

FILEID**RMSOCLOSE

M 4

RMS
VOL

RRRRRRRR	MM	MM	SSSSSSSS	000000	CCCCCCCC	LL	000000	SSSSSSSS	EEEEEEEEE
RRRRRRRR	MM	MM	SSSSSSSS	000000	CCCCCCCC	LL	000000	SSSSSSSS	EEEEEEEEE
RR	RR	MMMM	MMMM	SS	00	00	CC	00	SS
RR	RR	MMMM	MMMM	SS	00	00	CC	00	SS
RR	RR	MM	MM	MM	SS	00	0000	SS	EE
RR	RR	MM	MM	MM	SS	00	0000	SS	EE
RRRRRRRR	MM	MM	SSSSSS	00	00	CC	00	SS	EE
RRRRRRRR	MM	MM	SSSSSS	00	00	CC	00	SS	EE
RR	RR	MM	MM	SS	0000	00	CC	00	SS
RR	RR	MM	MM	SS	0000	00	CC	00	SS
RR	RR	MM	MM	SS	00	00	CC	00	SS
RR	RR	MM	MM	SS	00	00	CC	00	SS
RR	RR	MM	MM	SSSSSS	000000	CCCCCCC	LLLLLLLL	000000	SSSSSSSS
RR	RR	MM	MM	SSSSSS	000000	CCCCCCC	LLLLLLLL	000000	SSSSSSSS

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS

(3)	172	DECLARATIONS
(4)	215	RMSSCLOSE, \$CLOSE Routine
(7)	449	RMSCLSCU, Cleanup IFAB and Exit RMS
(8)	481	RMSRETIFB, Return IFAB but Leave File Open
(9)	503	RMSCLEANUP, Cleanup IFAB and Associated Storage
(11)	679	RMSSPL SCF - \$CLOSE routine for spool/submit options
(12)	824	RMSRELEASEALL, Release all BDB's

0000 1 \$BEGIN RMSOCLOSE,000,RMSRMS,<DISPATCH FOR CLOSE OPERATION>
0000 2
0000 3 :*****
0000 4 :*****
0000 5 :*
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :*
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :*
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :*
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :*
0000 24 :*
0000 25 :*****
0000 26 :

0000 28 :++
0000 29 :
0000 30 : Facility: rms32
0000 31 :
0000 32 : Abstract:
0000 33 : this module is the highest level control routine
0000 34 : to perform the \$close function.
0000 35 :
0000 36 : Environment: star processor running starlet exec.
0000 37 :
0000 38 :
0000 39 : Author: L F Laverdure, creation date: 1-MAR-1977
0000 40 :
0000 41 : Modified By:
0000 42 :
0000 43 : V03-033 SHZ0010 Stephen H. Zalewski 04-May-1984
0000 44 : Do not recreate address space in rm\$unmap_gbl because
0000 45 : the space is now taken directly from P0 space.
0000 46 :
0000 47 : V03-032 JEJ0011 J E Johnson 20-Mar-1984
0000 48 : Include global buffer quota accounting.
0000 49 :
0000 50 : V03-031 JEJ0020 J E Johnson 28-Mar-1984
0000 51 : Correct multiple problems in RAS270.
0000 52 :
0000 53 : V03-030 RAS0270 Ron Schaefer 14-Mar-1984
0000 54 : Remove the NAM block dependency for the SPL/SCF/DLT on
0000 55 : \$CLOSE options. Eliminate RMSCLOSE1.
0000 56 :
0000 57 : V03-029 DGB0011 Donald G. Blair 01-Mar-1984
0000 58 : Change the way the ACP is called as part of the restructuring
0000 59 : necessary for access mode protected files.
0000 60 :
0000 61 : V03-028 JWT0160 Jim Teague 29-Feb-1984
0000 62 : Remove calls to RMSDEALLEFN.
0000 63 :
0000 64 : V03-027 SHZ0009 Stephen H. Zalewski 12-Sep-1983
0000 65 : Change the sense of a branch so that the NWA DOES get
0000 66 : deallocated during the close.
0000 67 :
0000 68 : V03-026 SHZ0008 Stephen H. Zalewski 10-Aug-1983
0000 69 : Set a bit in the GBSB after we have decremented the accessor
0000 70 : count in the global buffer section (and possibly flushed the
0000 71 : cache as well) to prevent last chance
0000 72 : from decrementing the count again in the case where the
0000 73 : process gets stopped before we have completely cleaned up.
0000 74 :
0000 75 : V03-025 SHZ0007 Stephen H. Zalewski 02-Aug-1983
0000 76 : If last accessor to a global buffer section, then zero
0000 77 : the global section size, and global buffer count fields
0000 78 : in the lock value block for the section. This is to prevent
0000 79 : the case where the next accessor takes a lock before we
0000 80 : are done cleaning up, thus causing this subsequent accessor
0000 81 : to get incorrect data in the value block.
0000 82 :
0000 83 : V03-024 KBT0567 Keith B. Thompson 28-Jul-1983
0000 84 : Check for the NWA pointer not the flag

0000 85 :
0000 86 :
0000 87 : V03-023 DAS0001 David Solomon 20-Jul-1983
0000 88 : IFB\$V_JNL moved from IFB\$B_JNLFLG to IFB\$B_JNLFLG2.
0000 89 :
0000 90 : V03-022 KPL0003 Peter Lieberwirth 20-Jun-1983
0000 91 : Correct commentary regarding AT journaling of \$CLOSE.
0000 92 :
0000 93 : V03-021 SHZ0006 Stephen H. Zalewski 20-Jun-1983
0000 94 : Add code for cluster global buffers.
0000 95 :
0000 96 : V03-020 KPL0002 Peter Lieberwirth 31-May-1983
0000 97 : Write CLOSE mapping journal entry before buffers are
0000 98 : deallocated.
0000 99 :
0000 100 : V03-019 KBT0520 Keith B. Thompson 23-May-1983
0000 101 : The routine RMSCHKNAMBLK no longer exist and remove
0000 102 : the RMSCLS_CHKNAME hack.
0000 103 :
0000 104 : V03-018 KBT0509 Keith B. Thompson 5-May-1983
0000 105 : Use RMSDEALLOCATE_FWA
0000 106 :
0000 107 : V03-017 KPL0001 Peter Lieberwirth 29-Apr-1983
0000 108 : Always look for IFAB journaling structures to deallocate.
0000 109 :
0000 110 : V03-016 JWH0200 Jeffrey W. Horn 22-Mar-1983
0000 111 : Back out JWH0102, \$CLOSE now allowed within an active
0000 112 : RU.
0000 113 :
0000 114 : V03-015 KBT0457 Keith B. Thompson 10-Jan-1983
0000 115 : Deallocate fwa along with ifab.
0000 116 :
0000 117 : V03-014 SHZ0005 Stephen H. Zalewski, 19-Oct-1982 16:32
0000 118 : Remove code that checks for corrupt gbd's in a global
0000 119 : buffer section.
0000 120 :
0000 121 : V03-013 JWH0104 Jeffrey W. Horn 22-Sep-1982
0000 122 : Relocate calls to RMSMAPJNL, RMSDEAJNL to after deaccess.
0000 123 :
0000 124 : V03-012 JWH0102 Jeffrey W. Horn 22-Sep-1982
0000 125 : If the file is in a recovery-unit then don't allow the user
0000 126 : to close the file.
0000 127 :
0000 128 : V03-011 SHZ0004 Stephen H. Zalewski, 10-Sep-1982 17:04
0000 129 : Remove all references to SFD, SIFB and FRB as they no longer
0000 130 : exist.
0000 131 :
0000 132 : V03-010 KBT0339 Keith B. Thompson 16-Sep-1982
0000 133 : Always try to deallocate the gbsb (i.e. take it out of unmap)
0000 134 :
0000 135 : V03-009 SHZ0003 Stephen H. Zalewski, 7-Sep-1982 22:24
0000 136 : Add TEMPORARY code that checks for corruption of GBDs
0000 137 : in global buffer section.
0000 138 :
0000 139 : V03-008 SHZ0002 Stephen H. Zalewski, 7-Sep-1982 22:24
0000 140 : Move dequeuing of global buffer section lock so that when
0000 141 : a global section is unmapped the lock is immediately dequeued.
0000 141 : This prevents a second process from initializing a new section
0000 141 : incorrectly.

0000 142 :		
0000 143 :	V03-008 SHZ0002	Stephen H. Zalewski, 1-Sep-1982 15:37
0000 144 :	If stream has global buffers, then dequeue the lock it	
0000 145 :	had on the global buffer section, and remove its GBSB.	
0000 146 :		
0000 147 :	V03-007 KBT0179	Keith B. Thompson 23-Aug-1982
0000 148 :	Reorganize psects and rename entry point to single '\$'	
0000 149 :		
0000 150 :	V03-006 KBT0110	Keith B. Thompson 16-Jul-1982
0000 151 :	Deallocate the sfsb at a more appropriate time	
0000 152 :		
0000 153 :	V03-005 TMK0001	Todd M. Katz 02-Jul-1982
0000 154 :	Implement the RMS cluster solution for next record positioning.	
0000 155 :	As the next record context is now kept locally in the IRAB	
0000 156 :	instead of in individual NRP cells, there is no NRP list	
0000 157 :	to be returned.	
0000 158 :		
0000 159 :	V03-004 KDM0002	Kathleen D. Morse 28-Jun-1982
0000 160 :	Added \$PRDEF.	
0000 161 :		
0000 162 :	V03-003 JWH0002	Jeffrey W. Horn 19-May-1982
0000 163 :	Add support for journaling; Write out closing mapping entry	
0000 164 :	and then call RM\$DEAJNL.	
0000 165 :		
0000 166 :	V03-002 SHZ0001	Stephen H. Zalewski, 8-Jun-1982 17:02
0000 167 :	Remove instruction that cleared kernel mode flag after the	
0000 168 :	SIFAB was released (moved to RM\$SHARE).	
0000 169 :		
0000 170 :--		

0000 172 .SBTTL DECLARATIONS
0000 173
0000 174 :
0000 175 :: Macros:
0000 176 :
0000 177
0000 178 \$BDBDEF
0000 179 \$BLBDEF
0000 180 \$DEVDEF
0000 181 \$FABDEF
0000 182 \$FIBDEF
0000 183 \$FWADEF
0000 184 \$GBDDEF
0000 185 \$GBHDEF
0000 186 \$GBPBDEF
0000 187 \$GBSBDEF
0000 188 \$IFBDEF
0000 189 \$IMPDEF
0000 190 \$IODEF
0000 191 \$IRBDEF
0000 192 \$LCKDEF
0000 193 \$NAMDEF
0000 194 \$PSLDEF
0000 195 \$RABDEF
0000 196 \$RLBDEF
0000 197 \$RLSDEF
0000 198 \$RMSDEF
0000 199 \$SJCDDEF
0000 200
0000 201 :
0000 202 :: Equated Symbols:
0000 203 :
0000 204
0000 205 FOP=FAB\$L_FOP*8
0000 206 BKP=IRB\$L_BKPBITS*8
0000 207
0000 208 :
0000 209 :: Own Storage:
0000 210 :
0000 211
0000 212 MASK = ^X1FF
0000 213 : mask for getting to page boundary

```
0000 215 .SBTTL RMSSCLOSE, $CLOSE Routine
0000 216 :++
0000 217 :
0000 218 : RMSS$CLOSE - highest level file close routine
0000 219 :
0000 220 : this routine performs the highest level $close processing.
0000 221 : its functions include:
0000 222 :
0000 223 : 1. common setup
0000 224 : 2. check for all streams idle, exiting if not
0000 225 : 3. force a disconnect for all streams
0000 226 : 4. dispatch to organization-dependent code
0000 227 : 5. if the dlt fop bit is set and neither spl nor scf is set,
0000 228 :     delete the file
0000 229 : 6. return all bdb's and buffers
0000 230 : 7. deaccess the file if accessed
0000 231 : 8. return the asb and nwa (if any), the ifab, and all pages used for
0000 232 :     rms internal structures for this file
0000 233 : 9. zero the ifab table pointer and fab$w_ifi
0000 234 : 10. exit to the user, generating an ast if requested
0000 235 :
0000 236 :
0000 237 : Calling sequence:
0000 238 :
0000 239 : entered from exec as a result of user's calling sys$close
0000 240 : (e.g., by using the $close macro).
0000 241 :
0000 242 : Input Parameters:
0000 243 :
0000 244 :     ap      user's argument list addr
0000 245 :
0000 246 : Implicit Inputs:
0000 247 :
0000 248 :     the contents of the fab and possible related user interface
0000 249 :     blocks.
0000 250 :
0000 251 : Output Parameters:
0000 252 :
0000 253 :     r0      status code
0000 254 :     r1      destroyed
0000 255 :
0000 256 : Implicit Outputs:
0000 257 :
0000 258 :     the ifab and all related internal rms structures are vaporized.
0000 259 :     fab$l_sts and fab$l_stv are output and fab$w_ifi is zeroed if the
0000 260 :     close was successful.
0000 261 :
0000 262 :     a completion ast is queued if so specified by the user.
0000 263 :
0000 264 : Completion Codes:
0000 265 :
0000 266 :     standard rms
0000 267 :
0000 268 : Side Effects:
0000 269 :
0000 270 :     none
0000 271 :
```

```

0000 272 :--
0000 273 :--  

0000 274 SENTRY RMSSCLOSE
0000 275 STSTPT CLOSE
FFF7' 30 0006 276 BSBW RMSFSET ; do common setup
0009 277 ; note: does not return on error
0009 278
0009 279 :++  

0009 280 : force a disconnect on all streams
0009 281 :
0009 282
01  DD 0009 283 PUSHL #1 ; status code to stack
3B  11 000B 284 BRB NXTIRAB ; go check if any irabs linked
000D 285
000D 286 :
000D 287 : do an effective bsb to the org-specific disconnect
000D 288 :
000D 289 : note: this makes the fab look like a rab, but is of no consequence
000D 290 : since there are no rab inputs to the internal disconnect
000D 291 : and the only outputs are stv and isi (zeroed)
000D 292 :
000D 293
000D 294 ASSUME RABSL_STV EQ FABSL_STV
000D 295 ASSUME RABSW_ISI EQ FABSWIFI
000D 296 ASSUME RABSC_BLN LE FABSC_BLN ; (necessary for the re-probe on stall)
000D 297
000D 298 :
000D 299 : must clear the async operation bit to avoid arglist copy to asb, set busy,
000D 300 : and initialize ppf_image bit correctly
000D 301 :--  

000D 302 :
000D 303
000D 304 NXTDISC:  

04  0C  8A 000D 305 BICB #<1@<!RB$V_ASYNC-BKP>>!<1@<!RB$V_PPF_IMAGE-BKP>>,-
04 A9 000F 306 IRBSL_BKPBITS(R9)
0A A9 57 90 0015 307 SSB #IRBSV_BUSY,(R9) ; say rab busy
04 6A 22 E1 0019 308 MOVB R7,IRBSB_MODE(R9) ; set mode into irab
0280 8F 88 0021 309 BBC #IFBSV_PPF_IMAGE,(R10),10$ ; branch unless indirect ppf
36'AF 9F 0025 310 SSB #IRBSV_PPF_IMAGE,(R9) ; say irab accessed indirectly
0028 311 10$: PUSHR #^M<R7,R9> ; save mode and irab addr
0028 312 PUSHAB B^NXTRTN ; return pc to stack
0028 313 CASE TYPE=B, SRC=IFBSB_ORGCASE(R10),-
0028 314 DISPLIST=<RM$DISCONNECT1,RMSDISCOMMONSUC,RMSDISCONNECT3>
0280 8F BA 0033 315 BRW RM$DISCOMMONSUC ; handle unknown org (blk i/o)
0036 316 NXTRTN: POPR #^M<R7,R9> ; restore mode and (deallocated) irab addr
003A 317 : (link still valid)
04 6A 22 E1 003A 318 BBC #IFBSV_PPF_IMAGE,(R10),10$ ; branch unless indirect ppf
003E 319 CSB #IRBSV_BUSY,(R9) ; say irab not busy anymore
03 50 E8 0042 320 10$: BLBS R0,NXTIRAB ; branch if no error
6E 50 D0 0045 321 MOVL R0,(SP) ; replace status code
0048 322
0048 323 ASSUME IFBSL_IRAB_LNK EQ IRBSL_IRAB_LNK
0048 324
0048 325 NXTIRAB:  

59  1C A9  D0 0048 326 MOVL IRBSL_IRAB_LNK(R9),R9 ; get next irab
004C 327
004C 328 :*****
```

```

004C 329
004C 330 : note: the next irab link must still be good even though previous irab
004C 331 : is deallocated, since nothing else could have re-used the space.
004C 332
004C 333 :****
004C 334
      59  BF  12 004C 335     BNEQ   NXTDISC          ; loop if more irabs
      5A  5A  00 004E 336     MOVL    R10,R9           ; restore ifab address
0051
0051 338 :++
0051 339
0051 340 : get or of fab options that are input to either open/create or close
0051 341
0051 342 :--
0051 343
      12 69  22  E0 0051 344     BBS    #IFBSV_PPF_IMAGE,(R9),10$ ; branch if indirect ppf
0055 345
0055 346     ASSUME FABSV_RWC+1    EQ    FABSV_DMO
0055 347     ASSUME FABSV_DMO+1    EQ    FABSV_SPL
0055 348     ASSUME FABSV_SPL+1    EQ    FABSV_SCF
0055 349     ASSUME FABSV_SCF+1    EQ    FABSV_DLT
0055 350
      51  68  05  28  EF 0055 351     EXTZV #FABSV_RWC+FOP,#5,(R8),R1 ; get option bits from fab
005A 352
005A 353     ASSUME IFBSV_RWC+1    EQ    IFBSV_DMO
005A 354     ASSUME IFBSV_DMO+1    EQ    IFBSV_SPL
005A 355     ASSUME IFBSV_SPL+1    EQ    IFBSV_SCF
005A 356     ASSUME IFBSV_SCF+1    EQ    IFBSV_DLT
005A 357
      50  69  05  27  EF 005A 358     EXTZV #IFBSV_RWC,#5,(R9),R0 ; get saved ifab copies from $open
      50  50  51  88  88 005F 359     BISB2 R1,R0           ; or them together
      69  05  27  50  F0 0062 360     INSV   R0,#IFBSV_RWC,#5,(R9) ; and restore in ifab flags
0067 361
0067 362 :++
0067 363
0067 364 : dispatch to organization-dependent close code
0067 365
0067 366 : register state for dispatch:
0067 367
0067 368     r11    impure area address
0067 369     r10    ifab address
0067 370     r9     ifab address
0067 371     r8     fab address
0067 372     (sp)   return address
0067 373     4(sp)  status code
0067 374
0067 375 :--
0067 376
0067 377     PUSHAB B^CLSDLT          ; return pc to stack
0067 378     CASE   TYPE=B,SRC=IFBSB ORGCASE(R9),-
0067 379     DISPLIST=<RM$NULL,RM$NULL,RMSCLOSE3> ; pick up correct routine
0067 380     TSTL   (SP)+           ; remove return pc for other orgs
0067 381
0067 382 :++
0067 383
0067 384     NOTE: Since there is only a special close routine for isam make life
0067 385     a little simpler. If the above code is ever used the call to

```

0067 386 : rm\$close3 will have to be done through rm3face since the word
0067 387 : branch of the case will not reach the real rm\$close3
0067 388 :
0067 389 :--
0067 390 :
23 A9 02 91 0067 391 10\$: CMPB #IFB\$C_IDX,IFB\$B_ORGCASE(R9) ; Is this isam
06 12 0068 392 BNEQ CLSDLT ; Branch if not
00000000'EF 16 006D 393 JSB RM\$CLOSE3 ; Do the close
0073 394
0073 395

```

0073 397
0073 398 ;++
0073 399
0073 400 ; return here from organization-dependent close routines
0073 401
0073 402 ;--
0073 403
0073 404 ; check for dlt fop bit set.
0073 405 ; if set and the spl and scf bits are clear, delete the file.
0073 406
0073 407 ;--
0073 408
47 69 2B E1 0073 409 CLSDLT: BBC #IFBV_DLT,(R9),CLSCU1 ; branch if dlt not speced
0077 410
0077 411
0077 412 ; if this is a network operation, do not process dlt option; it will be
0077 413 ; handled by network code during deaccess.
0077 414
0077 415
69 0D 43 E0 0077 416 BBS #DEV$V_NET,IFBSL_PRIM_DEV(R9),- ; branch if network operation
007A 417 CLSCU1
007B 418
007B 419 ASSUME IFBV_SCF EQ IFBV_SPL+1
007B 420
00 69 02 29 ED 007B 421 CMPZV #IFBV_SPL,#2,(R9),#0 ; spl and scf both 0?
3C 12 0080 422 BNEQ CLSCU1 ; branch if not
69 1C E1 0082 423 BBC #DEV$V_RND,IFBSL_PRIM_DEV(R9),- ; ignore if not disk
38 0085 424 CLSCU1
69 18 E0 0086 425 BBS #DEV$V_FOR,IFBSL_PRIM_DEV(R9),- ; ignore if mntd foreign
34 0089 426 CLSCU1
30 69 22 E0 008A 427 BBS #IFBV_PPF_IMAGE,(R9),CLSCU1 ; ignore if indirect ppf
008E 428
008E 429 ; build fib for delete
008E 430
008E 431
008E 432
5A 38 A9 D0 008E 433 BLDFIB: MOVL IFBSL_FWA_PTR(R9),R10 ; get FWA addr
51 14 AA D0 0092 434 MOVL FWASQ_FIB#4(R10),R1 ; get FIB addr
05 69 35 E0 0096 435 BBS #IFBV_TMP,(R9),DELETE ; branch if no directory entry
009A 436 SSB #IFBV_FINDFID,FIBSW_NMCLL(R1) ; indicate find via fid
009F 437 ; (0 fid => current file)
50 0135 8F 3C 009F 438 DELETE: MOVZWL #<IOS_DELETE!IOSM_DELETE>,R0 ; set i/o func code
7E 7C 00A4 439 CLRQ -(SP) ; p6 = p5 = 0 for qio
7E 7C 00A6 440 CLRQ -(SP) ; p4 = p3 = 0 for qio
7E D4 00A8 441 CLRL -(SP) ; p2 = 0 for qio
FF53 30 00AA 442 BSBW RMSFCPFNC ; call acp to do the delete
OE 50 E8 00AD 443 BLBS R0,CLSCU1 ; branch if okay
00B0 444 RMSERR MKD(SP) ; replace error code
OC A8 OC A9 D0 00B5 445 MOVL IFBSL_IOS(R9),FABSL_STV(R8) ; return system error code
02 11 00BA 446 BRB CLSCUT ; return to common close
00BC 447

```

00BC 449 .SBTTL RM\$CLSCU, Cleanup IFAB and Exit RMS
00BC 450
00BC 451 :++
00BC 452 : RM\$CLSCU - clean part of close operation for error paths
00BC 453 :
00BC 454 : entry point to clean up on aborted \$open or \$create
00BC 455 : or successful \$erase,\$sparse or other control routine
00BC 456 :
00BC 457 :--
00BC 458
00BC 459 RM\$CLSCU::
50 DD 00BC 460 PUSHL R0 ; save error code
00BE 461
00BE 462 :
00BE 463 : close of indirectly opened process permanent file.
00BE 464 : need merely zero the ifi field.
00BE 465 :
00BE 466
09 69 22 E1 00BE 467 CLSCU1: BBC #IFBSV_PPF_IMAGE,(R9),20\$; branch if not indirect ppf
02 A8 B4 00C2 468 CLRW FABSW_IFI(R8)
50 8ED0 00C5 469 POPL R0 ; restore status code
FF35' 31 00C8 470 BRW RMSEXRMS ; all set
00CB 471
00CB 472 :
00CB 473 : cleanup ifab and all associated structures
00CB 474 :
00CB 475 :
03 11 10 00CB 476 20\$: BSBB RM\$CLEANUP ; cleanup ifab and stuff
50 E9 00CD 477 BLBC R0,CLS_EX ; branch if cleanup error
50 8ED0 00D0 478 POPL R0
FF2A' 31 00D3 479 CLS_EX: BRW RMSEX_NOSTR ; and do structureless exit

00D6	481	.SBTTL RMSRETIFB, Return IFAB but Leave File Open		
00D6	482			
00D6	483	:++		
00D6	484			
00D6	485	RMSRETIFB - evaporate internal structures but leave accessed		
00D6	486			
00D6	487	Entry point to leave file open for user but remove all rms		
00D6	488	knowledge of the file		
00D6	489			
00D6	490	outputs:		
00D6	491			
00D6	492	r0 = status code		
00D6	493			
00D6	494	:--		
00D6	495			
00D6	496	RMSRETIFB::		
FA AF	9F	00D6	497	PUSHAB B^CLS_EX ; return address from cleanup
50	DD	00D9	498	PUSHL R0 ; save status code
0075	31	00DB	499	BRW CLNJNL ; only return ifab, etc.
		00DE	500	
		00DE	501	; by jumping into cleanup

```

      00DE 503      .SBTTL RMSCLEANUP, Cleanup IFAB and Associated Storage
      00DE 504
      00DE 505 :++
      00DE 506
      00DE 507      RMSCLEANUP - deallocate internal structures
      00DE 508
      00DE 509      Cleanup the ifab and its associated structures (bdb's, etc.)
      00DE 510      if an access has been done, a deaccess is performed. any
      00DE 511      outstanding channels are deassigned. the ifi entry in the
      00DE 512      ifab table is deleted.
      00DE 513
      00DE 514      inputs:
      00DE 515
      00DE 516      r9 = ifab address
      00DE 517      r11 = impure area
      00DE 518
      00DE 519      outputs:
      00DE 520
      00DE 521      r0 = status code
      00DE 522      r7 = caller's access mode
      00DE 523      ap = restored argument list pointer
      00DE 524
      00DE 525      (r7,ap are used as inputs to rm$ex_nostr)
      00DE 526      --
      00DE 527
      00DE 528
      00DE 529      RMSCLEANUP:::
      00DE 530
      00DE 531      :
      00DE 532      preset status to be returned to caller
      00DE 533
      00DE 534
      01    DD 00DE 535      PUSHL #1                      : preset status to return
      01    E1 00EO 536
      0C 00A2 C9 00E0 537      BBC      #IFBSV_JNL,-          :
      00000000'EF 16 00E2 538      IFBSB JNLFLG2(R9),20$   : branch if no journaling or if an
      03 50       E8 00E6 539      JSB      RMSMAPJNL           : error turned journaling off
      6E 50       D0 00EF 540      BLBS     R0,10$             : write out $CLOSE mapping entries
      00F2        541      MOVL     R0,(SP)            : branch on success
      00F2        542      10$:                           : otherwise replace status code
      00F2        543
      00F2        544
      00F2        545      : return all buffers and bdb's
      00F2        546
      00F2        547
      0148   30 00F2 548 20$:      BSBW     RMSRELEASALL
      00F5        549
      00F5        550
      00F5        551      : deaccess the file
      00F5        552
      00F5        553
      00F5        554      DEACCESS:
      22 69 25. E5 00F5 555      BBCC     #IFBSV_ACCESSION,(R9),10$ ; branch if file not accessed
      FF04. 30 00F9 556      BSBW     RMSDEACCESS           ; do the deaccess
      03 50       E8 00FC 557      BLBS     R0,5$              ; branch on success
      6E 50       D0 00FF 558      MOVL     R0,(SP)            ; save error code
      0102
      0102 559

```

0102 560 :
 0102 561 : if this is a network operation, do not process scf and spl options by
 0102 562 : rm\$close1; these will be handled by network code during deaccess.
 0102 563 :
 0102 564 :
 15 69 0D E0 0102 565 \$S: BBS #DEV\$V_NET,IFB\$L_PRIM_DEV(R9),10\$; branch if network operation
 0106 566 :
 0106 567 ASSUME FABSC_SEQ EQ 0
 23 A9 95 0106 568 TSTB IFB\$B_ORGCASE(R9) : sequential file org?
 00 69 02 29 10 12 0109 569 BNEQ 10\$: branch if not
 09 13 0110 570 CMPZV #IFB\$V_SPL,#2,(R9),#0 : spl and scf both 0?
 0086 30 0112 571 BEQL 10\$: branch if so
 03 50 E8 0115 572 BSBW RMS\$SPL_SCF : check for spool or submit
 6E 50 D0 0118 573 BLBS R0,10\$: branch on success
 0118 574 MOVL R0,(SP) : save error code
 1F 69 14 E0 0118 575 :
 25 69 05 E1 011F 576 10\$: BBS #DEV\$V_MBX,IFB\$L_PRIM_DEV(R9),CLSMLBX ; branch if mailbox
 0123 577 BBC #DEV\$V_SQD,IFB\$L_PRIM_DEV(R9),DEASSIGN ; branch if not magtape
 0123 578 :
 0123 579 :
 0123 580 : foreign magtape - write end of tape if ifb\$v_eof is set
 0123 581 : rewind if rwc (rewind on close) is set
 0123 582 :
 0123 583 :
 0123 584 :
 0123 585 CLSMAGTAP:
 21 69 18 E1 0123 586 BBC #DEV\$V_FOR,IFB\$L_PRIM_DEV(R9),DEASSIGN ; branch if not foreign
 FED6' 30 0127 587 BSBW RMSWTTPMARK : write tape marks
 18 50 E9 012A 588 BLBC R0,DEASSIGN_ALT : go away if error
 012D 589 :
 17 69 27 E1 012D 590 BBC #IFB\$V_RWC,(R9),DEASSIGN ; branch if no rewind
 5A DD 0131 591 PUSHL R10 : rewind wants ifab in r10
 5A 59 DO 0133 592 MOVL R9,R10 : r10 <- ifab
 FEC7' 30 0136 593 BSBW RM\$REWIND_MT : do rewind
 5A 8ED0 0139 594 POPL R10 : restore r10
 07 11 013C 595 BRB DEASSIGN_ALT : join mainstream
 013E 596 :
 013E 597 :
 013E 598 : mail box - write end of file if write access was allowed
 013E 599 :
 013E 600 :
 013E 601 CLSMLBX:
 06 69 30 E1 013E 602 BBC #IFB\$V_WRTACC,(R9),DEASSIGN ; branch if no write
 FEBB' 30 0142 603 BSBW RMSWRITEOF : write end of file
 0145 604 :
 6E 50 D0 0145 605 DEASSIGN_ALT:
 0145 606 MOVL R0,(SP) : save status
 0148 607 :
 0148 608 : deassign i/o channel
 0148 609 :
 0148 610 :
 0148 611 :
 0148 612 DEASSIGN:
 0148 613 \$DASSGN_S IFB\$W_CHNL(R9)
 0153 614 :

**F

			0153	616	:		
			0153	617	:	clean up journaling	
			0153	618	:		
			0153	619	:		
			0153	620	CLNJNL:		
00000000'EF	16	0153	621	JSB	RM\$DEAJNL	: clean up journaling	
06 50	E8	0159	622	BLBS	R0,RETIFB	: branch on success	
03 6E	E9	015C	623	BLBC	(SP),RETIFB	: branch if code already error	
6E 50	DO	015F	624	MOVL	R0,(SP)	: otherwise replace status code	
			0162	625			
			0162	626	:		
			0162	627	:	Return the ifab, sfsb, asb, fwa(s) and nwa if any	
			0162	628	:		
			0162	629	:		
			0162	630	RETIFB:		
5C 18 A9	DO	0162	631	MOVL	IFBSL_ARGLST(R9),AP	: restore arglist address	
57 0A A9	9A	0166	632	MOVZBL	IFBSB_MODE(R9),R7	and caller's mode	
FE93'	30	016A	633	BSBW	RMSRLS_SF_SFB	: clean up sfsb if any	
3C A9	D5	016D	634	TSTL	IFBSL_NWA_PTR(R9)	: check for nwa	
03	13	0170	635	BEQL	10\$: branch if nwa not present	
FE8B'	30	0172	636	BSBW	NTSNWA_FREE	: deallocate nwa	
54 14 A9	DO	0175	637	10\$:	MOVL	: get asb addr	
09	13	0179	638	BEQL	20\$: branch if none	
53 5B	DO	017B	639	MOVL	R11,R3	: ifab asb came from imp. free space header	
FE7F'	30	017E	640	BSBW	RMSRETBLK	: deallocate asb	
14 A9	D4	0181	641	CLRL	IFBSL_AS_BADDR(R9)	: clear pointer	
FE79'	30	0184	642	BSBW	RMSDEALLOCATE_FWA	: deallocate fwa and related structures	
53 59	7D	0187	643	MOVQ	R9,R3	: ifab addr to right regs for retblk	
FE73'	30	018A	644	BSBW	RMSRETBLK	: deallocate ifab	
			018D	645			
			018D	646	:		
			018D	647	:	Return the pages used for all internal structures	
			018D	648	:		
			018D	649	:		
5A 08	C2	018D	650	SUBL2	#8,R10	: get start of page addr	
54 6A	DO	0190	651	MOVL	(R10),R4	: get 1st hole (there must be	
000001F8 8F	56 64	DO	0193	652		at least 1 hole for ifab)	
08 A4	D1	0196	653	40\$:	MOVL	: get next free space hole	
			019E	654	CMPL	: all holes should now be equal	
			019E	655		to one page in length	
			019E	656		(less list head if page 1)	
			019E	657	BLSSU	: branch if not	
			FE5D'	30	01A0	ERRBUG	
			54 56	DO	01A3	RMSRET1PAG	
			5A 56	D1	01A6	MOVL	
			E8	12	01A9	R6,R4	
					660	: get set to return next page	
					CMPL	: all done?	
					R6,R10	: loop if not	
					BNEQ		
					40\$		
			01AB	662			
			01AB	663	:		
			01AB	664	:	Zero ifi and ifab table pointer	
			01AB	665	:		
			01AB	666			
			FE52'	30	01AB	BSBW RMSZAPIFI	: zero ifi and ifab table entry
			50 8ED0	01AE	667	POPL R0	: return status to caller
			05	01B1	668	RSB	: return to caller
					669		
					01B2		
					670		
					01B2		
					671		
					01B2		
					672	:	
						Attempted to return an ifab-related page having some	

RMSOCLOSE
V04-000

D 6
DISPATCH FOR CLOSE OPERATION 16-SEP-1984 01:11:09 VAX/VMS Macro V04-00
RM\$CLEANUP, Cleanup IFAB and Associated 5-SEP-1984 16:24:38 [RMS.SRC]RMSOCLOSE.MAR;1

Page 16
(10)

RM'
Tat

01B2 673 : non-deallocated block(s) in it
01B2 674 :
01B2 675 :
01B2 676 ERRBUG: RMSTBUG FTLS_DEALLERR
01B9 677 :

01B9 679 .SBTTL RMSSPL_SCF - \$CLOSE routine for spool/submit options
01B9 680
01B9 681 :++
01B9 682 : RMSSPL_SCF - specific close code for the SPL and SCF FOP options
01B9 683 :
01B9 684 : This routine performs the spl and scf options, and if set (either on
01B9 685 : \$open/\$create or on \$close), sends a message to the job controller to
01B9 686 : queue the file to the sys\$print or sys\$batch queues respectively.
01B9 687 : If both spl and scf are set, scf takes precedence. The dlt fop sub-option
01B9 688 : is passed on to the job controller.
01B9 689 :
01B9 690 : The overall flow of the routine is as follows:
01B9 691 :
01B9 692 : 1. build the dvi, did and fid fields from the fwa onto the stack.
01B9 693 : 2. allocate a buffer on the stack to build the item list for the job
01B9 694 : controller.
01B9 695 : 3. fill the queue name in the appropriate item; the job controller will
01B9 696 : translate either SYSSPRINT or SYSSBATCH.
01B9 697 : 4. point an item at the dvi, did, fid copy on th stack
01B9 698 : 5. fill in the delete option if required.
01B9 699 : 6. send the message to the job controller with a function code of
01B9 700 : sjc\$_queue.
01B9 701 :
01B9 702 : Calling sequence:
01B9 703 :
01B9 704 : E W RMSSPL_SCF
01B9 705 :
01B9 706 : Input Parameters:
01B9 707 :
01B9 708 : r10 ifab address
01B9 709 : r9 ifab address
01B9 710 : r8 fab address
01B9 711 :
01B9 712 : Implicit Inputs:
01B9 713 :
01B9 714 : the contents of the ifab (especially ifb\$v_spc, scf, and dlt)
01B9 715 : the contents of the fwa (especially fwa\$q_shrfil and fwa\$st_fibbuf)
01B9 716 :
01B9 717 : Output Parameters:
01B9 718 :
01B9 719 : r1-r7 destroyed
01B9 720 : r0 status code
01B9 721 :
01B9 722 : Implicit Outputs:
01B9 723 :
01B9 724 : fab\$1_stv is set to subsidiary error code on an error.
01B9 725 :
01B9 726 : Completion Codes:
01B9 727 :
01B9 728 : standard rms, in particular, spl.
01B9 729 :
01B9 730 : Side Effects:
01B9 731 :
01B9 732 : none
01B9 733 :
01B9 734 : Note: no need to check that PPF_IMAGE not set since can't get here if so.
01B9 735 :

01B9 736 : job controller item list is currently 10 longwords
 01B9 737 : long - for three items and terminator
 01B9 738 :
 01B9 739 :--
 01B9 740 :
 01B9 741 : Own Storage:
 01B9 742 :
 01B9 743 :
 01B9 744 :
 54 4E 49 52 50 24 53 59 53 01B9 745 SYSPRINT: .ASCII /SYSSPRINT/
 00000009 01C2 746 SYSPRINT_LEN = .-SYSPRINT
 48 43 54 41 42 24 53 59 53 01C2 747 SYSBATCH: .ASCII /SYSSBATCH/
 00000009 01CB 748 SYSBATCH_LEN = .-SYSBATCH
 0000001C 01CB 749 ID_SIZE = 16 + 6 + 6 : DVI + DID + FID
 01AE 0F 00 61 50 2C 01CB 750 :
 50 57 SE 38 A9 D0 01CE 751 RMSSPL_SCF::
 0190 C7 7D 01D2 752 :
 6E 50 90 01D7 753 : Allocate DVI_DID_FID buffer on stack and fill it in.
 01AE 0F 00 61 50 2C 01DA 754 :
 01E1 755 :
 01E1 756 :
 01E1 757 :
 01E1 758 :
 01E1 759 SUBL2 #<ID_SIZE>,SP
 01E1 760 MOVL IFBS[FWA_PTR(R9),R7 : get FWA ptr
 01E1 761 MOVQ FWASQ_SHRFIL(R7),R0 : get DVI description
 01E1 762 MOVB R0,(SP) : make it ASCII
 01E1 763 MOVC5 R0,(R1),#0,#15,1(SP) : copy dvi and fill to 16 bytes
 01E1 764 :
 01E1 765 ASSUME FIBSW_FID+6 EQ FIBSW_DID
 01E1 766 :
 63 01F8 C7 OC 28 01E1 767 MOVC3 #<6+6>,FWAST_FIBBUF+FIBSW_FID(R7),(R3)
 01E1 768 : copy did and fid
 52 5E D0 01E7 769 MOVL SP,R2 : remember addr of dvi_did_fid blk
 01EA 770 : R3 points to cleaned-off SP
 01EA 771 :
 01EA 772 : Build the job controller item list
 01EA 773 :
 01EA 774 :
 01EA 775 :
 08 69 7E D4 01EA 776 CLRL -(SP) : end-of-list flag
 28 E1 01EC 777 BBC #IFBSV_DLT,(R9),10\$: branch if no delete requested
 00180000 7E 7C 01FO 778 CLRQ -(SP) : no retlen or addr
 8F DD 01F2 779 PUSHL #SJCS_DELETE_FILE@16 : delete flag
 01F8 780 : and zero bufflen
 01F8 781 :
 01F8 782 :
 01F8 783 : Now, point to the dvi_did_fid block to identify the file
 01F8 784 :
 01F8 785 :
 0027001C 7E D4 01F8 786 10\$: CLRL -(SP) : no retlen
 52 DD 01FA 787 PUSHL R2 : addr of block
 8F DD 01FC 788 PUSHL #<<SJCS_FILE_IDENTIFICATION@16>+ID_SIZE> : item type
 0202 789 : and fill-in size of identification
 0202 790 :
 0202 791 :
 0202 792 : Fill in the initial item, which indicates to the job controller that a file is

0202 793 : to be queued to either SYSPRINT or SYSBATCH.
0202 794 :
0202 795 :
0202 796 ASSUME SYSPRINT_LEN EQ SYSBATCH_LEN
0202 797 CLRL -(SP)
05 69 7E D4 0202 798 BBS #IFBSV SCF,(R9),20\$; no retlen
AE AF 2A EO 0204 799 PUSHAL B^SYSPRINT ; branch if submit command file
03 11 DF 0208 800 BRB 30\$; point to queue name string
B2 AF DF 020D 802 20\$: PUSHAL B^SYSBATCH ; do next item
00860009 8F DD 0210 803 30\$: PUSHL #<>SJCS_QUEUE@16>+SYSPRINT_LEN ; point to queue name
0216 804 ; indicate function
0216 805 ; and fill in length of queue name
51 SE DO 0216 806 MOVL SP,R1 ; addr of itemlist
0219 807 :
0219 808 : Call the job controller.
0219 809 :
0219 810 :
0219 811 :
0219 812 SSNDJBC_S -
0219 813 EFN = #IMPSC_ASYQIOEFN,- ; throw-away event flag
0219 814 FUNC = #SJCS_ENTER_FILE,- ; function
0219 815 ITMLST = (R1) ; item list
0220 816 :
5E 53 DO 0220 817 MOVL R3,SP ; clean stack
09 50 09 50 E8 0230 818 BLBS R0,40\$; exit on error
OC AB 50 DO 0233 819 MOVL R0,FAB\$L_STV(R8) ; save jobctl status
0237 820 RMSERR SPL ; and report error
05 023C 821 40\$: RSB
023D 822 :

023D 824 .SBTTL RMS\$RELEASALL, Release all BDB's
 023D 825
 023D 826 :++
 023D 827 : RMS\$RELEASALL - release bdb's and buffers
 023D 828 :
 023D 829 : Subroutine to release all bdb's and their associated buffers.
 023D 830 : Assumes dirty buffers will not be found.
 023D 831 : Also return all BLB's.
 023D 832 :
 023D 833 : inputs:
 023D 834 : r11 impure area address
 023D 835 : r9 ifab address
 023D 836 : r8 fab address
 023D 837 :
 023D 838 : outputs:
 023D 839 : r10 ifab address
 023D 840 : r0-r6 destroyed
 023D 841 --
 023D 842 :
 023D 843 : return all buffers and bdb's
 023D 844 :
 023D 845 :--
 023D 846 :
 023D 847 RMS\$RELEASALL::
 56 5A 59 D0 023D 848 MOVL R9,R10 : make sure r10 = ifab addr
 56 40 AA DE 0240 849 MOVAL IFBSL_BDB_FLNK(R10),R6 ; get bdb list head
 54 66 D0 0244 850 10\$: MOVL (R6),R4 ; get 1st bdb in list
 56 54 D1 0247 851 CMPL R4,R6 ; back at list head?
 1E 13 024A 852 BEQL 30\$; branch if yes - all done
 51 0A A4 01 E0 024C 853 BBS #BDB\$V_DRT, BDB\$B_FLGS(R4), DRTBUG ; Don't expect to find dirt.
 OC A4 B5 0251 854 TSTW BDB\$W_USERS(R4) ; use count nonzero?
 03 12 0254 855 BNEQ 20\$; no, go release bdb
 OC A4 B6 0256 856 INCW BDB\$W_USERS(R4) ; make it look accessed
 FDA4' 30 0259 857 20\$: BSBW RMSRLNERR ; go release it and free buffer.
 025C 858
 025C 859 ASSUME BDB\$B_BID EQ GBP\$B_BID
 025C 860 ASSUME <BD\$C_BID@1> EQ 0
 025C 861 ASSUME <GBP\$C_BID@1> EQ 1
 025C 862
 05 08 A4 E8 025C 863 BLBS BDB\$B_BID(R4), 25\$; br if gpb.
 FD9D' 30 0260 864 BSBW RMS\$RET_BDB ; return the bdb
 DF 11 0263 865 BRB 10\$; keep going until all gone.
 FD98' 30 0265 866 25\$: BSBW RMS\$RET_GPB ; return gpb.
 DA 11 0268 867 BRB 10\$; keep going
 026A 868
 026A 869 ASSUME IFBSW_AVGBPB EQ <IFBSW_AVLCL + 2>
 026A 870
 56 2C 0084 CA D4 026A 871 30\$: CLRL IFBSW_AVLCL(R10) ; Note all buffers gone.
 6A 33 E0 026E 872 BBS #IFBSV_NORECLK, (R10), R6 RA-EX ; All done if no locking.
 0098 CA DE 0272 873 MOVAL IFBSL_BLBFLNK(R10), R6 ; Get list head for BLB's.
 54 66 D0 0277 874 40\$: MOVL (R6),R4 ; Get next BLB.
 56 54 D1 027A 875 CMPL R4,R6 ; Back at list head?
 12 13 027D 876 BEQL CHKGBL ; All done then.
 24 A4 D5 027F 877 TSTL BLBSL_LOCK_ID(R4) ; This one still locked?
 08 13 0282 878 BEQL 45\$; EQL no lock, so just return it.
 FD79' 30 0284 879 BSBW RMSRLNERR ; Release the lock first.
 54 009C CA D0 0287 880 MOVL IFBSL_BLBBLNK(R10), R4 ; Recover BLB address.

FD71' 30 028C 881 45\$: BSBW RMS\$RETBLB : Return the BLB.
E6 11 028F 882 BRB 40\$: Go get next one.
0291 883
0088 CA D5 0291 884 CHKGBL: TSTL IFBSL_GBH_PTR(R10) : Are global buffers present?
07 13 0295 885 BEQL RA_EX : No, we are done.
FD66' 30 0297 886 BSBW RMSRAISE_GBS_LOCK : Get EX lock on global section.
0D 10 029A 887 BSBB RMSRELEASE_GBL_BUFFERS : Release and cleanup global buffers.
6E 10 029C 888 BSBB RMSUNMAP_GBL : Disassociate from section.
FD5F' 30 029E 889 RA_EX: BSBW RMSRLS_GBSB : Deallocate the GBSB if any (also release L
05 02A1 890 RSB
02A2 891
02A2 892 DRTBUG: RMSTBUG FTLS_RLSVRT : A dirty buffer has been left
02A9 893
02A9 894 behind by someone.

02A9 896 :++
 02A9 897 :
 02A9 898 : RMSRELEASE_GBL_BUFFERS
 02A9 899 :
 02A9 900 : This routine decrements the access count for the global buffer section.
 02A9 901 : If the access count goes to zero, then all cached buffers are released by
 02A9 902 : dequeuing the system lock for each buffer, and the system file lock is
 02A9 903 : also released.
 02A9 904 :
 02A9 905 : As a part of releasing the system lock on a buffer, we also give back the
 02A9 906 : quota used when that lock was first converted. Notice that if this routine
 02A9 907 : is called as part of a \$CLOSE operation then there exists a non-closeable
 02A9 908 : hole in which we can give back the quota and have the process deleted before
 02A9 909 : dequeuing the lock. This will have the effect of increasing the global
 02A9 910 : buffer quota by one. The reverse can occur during conversion to the system
 02A9 911 : lock in RMRELEASE.
 02A9 912 :
 02A9 913 : Note: This routine assumes that an EX lock has already been taken on the
 02A9 914 : global section.
 02A9 915 :
 02A9 916 : Inputs:
 02A9 917 :
 02A9 918 : R10 - Address of ifab.
 02A9 919 :
 02A9 920 : Outputs:
 02A9 921 :
 02A9 922 : none
 02A9 923 :
 02A9 924 :--
 02A9 925 RMSRELEASE_GBL_BUFFERS::
 54 7E 53 7D 02A9 926 MOVQ R3,-(SP) : Save registers.
 53 0088 CA D0 02AC 927 MOVL IFBSL_GBH_PTR(R10),R4 : Get address of global section in R4.
 53 7C AA D0 02B1 928 MOVL IFBSL_GBSB_PTR(R10),R3 : Get gbsb address in R3.
 01 01 E0 02B5 929 BBS #GBSB\$M NOTACCESSED,- : If set then access count is already decrem
 07 0B A3 02B7 930 GBSBSB FLAGS(R3), \$S : go check access count (we are in last cha
 1C A4 D7 02BA 931 DECL GBHSL_OSECNT(R4) : Decrement accessor count.
 01 88 02BD 932 BISB2 #GBSB\$M NOTACCESSED,- : Set bit in GBSB saying accessor count
 0B A3 02BF 933 GBSBSB FLAGS(R3) : has been decremented (for last chance)
 1C A4 D5 02C1 934 5\$: TSTL GBHSL_OSECNT(R4) : Test accessor count.
 42 12 02C4 935 BNEQ DONE : Exit if not last accessor.
 02C6 936 :
 53 54 D0 02C6 937 MOVL R4,R3 : Move address of section into R3.
 53 63 C0 02C9 938 10\$: ADDL2 (R3),R3 : Get address to next GBD element.
 54 53 D1 02CC 939 CMPL R3,R4 : Are we back at queue header?
 1F 13 02CF 940 BEQL RL\$ FILE LOCK : Yes, go release system file lock.
 14 A3 D5 02D1 941 TSTL GBD\$LOCK_ID(R3) : Is this buffer cached?
 F3 13 02D4 942 BEQL 10\$: No, go to next GBD.
 00000000'9F 01 58 02D6 943 ADAWI #1,2#RMSS\$GW GBLBUFQUO : Give the buffer back to the quota ctr
 14 A3 D4 02DD 944 SDEQ_S LKID = GBD\$LOCK_ID(R3) : DEQ the system lock on buffer.
 D9 11 02EB 945 CLRL GBD\$LOCK_ID(R3) : Mark this GBD as gone.
 02EE 946 BRB 10\$: Go to next GBD.
 02F0 947 :
 02F0 948 RLS_FILE_LOCK: :
 02F0 949 \$DEQ_S LKID = GBHSL_LOCK_ID(R4); \$DEQ system file lock.
 02FE 950 :
 54 7C AA D0 02FE 951 MOVL IFBSL_GBSB_PTR(R10),R4 : Get address of GBSB.
 38 A4 D4 0302 952 CLRL GBSBS\$GS_SIZE(R4) : Zero all fields in lock value

RMSOCLOSE
V04-000

DISPATCH FOR CLOSE OPERATION
RM\$RELEASEALL, Release all BDB's

K 6

16-SEP-1984 01:11:09 VAX/VMS Macro V04-00
5-SEP-1984 16:24:38 [RMS.SRC]RMSOCLOSE.MAR;1

Page 23
(13)

RM
VO

34 A4	B4	0305	953	CLRW	GBSB\$W_GBC(R4)	; block.
		0308	954			
53 8E	7D	0308	955	DONE:	MOVQ (SP)+,R3	; Restore registers.
	05	030B	956	RSB		
		030C	957			

030C 959 ;++
030C 960
030C 961 : RMSUNMAP_GBL
030C 962
030C 963 : This routine deletes the specified address range for the purpose of
030C 964 : un-mapping from a global section that has been used for i/o buffers.
030C 965
030C 966 : Note: This routine assume an EX lock is already held on the global section.
030C 967
030C 968 : Inputs:
030C 969
030C 970 : R0 - start address of range. (alt. entry pt.)
030C 971 : R1 - end address of range. (alt. entry pt.)
030C 972 : R10 - ifab address
030C 973
030C 974 : Outputs:
030C 975
030C 976 : Destroys R0 - R2.
030C 977
030C 978 :--
030C 979 RMSUNMAP_GBL::
51 50 0088 CA D0 030C 980 MOVL IFBSL_GBH_PTR(R10),R0 ; Put address of global section in R0.
50 10 A0 C1 0311 981 ADDL3 GBHSL_GS_SIZE(R0),R0,R1 ; End addr of sec + 1
51 D7 0316 982 DECL R1 ; End addr of section.
0318 983
7E 50 7D 0318 984 RMSUNMAP_GBL_ALT::
52 5E D0 031B 985 MOVQ R0, -(SP) ; Save range on stack.
031E 986 MOVL SP, R2 ; Remember that address.
50 8E 7D 032B 987 \$DELTVA_S INADR=(R2) ; Delete the VA.
05 032E 988 MOVQ -(SP)+, R0 ; Return address array.
032F 989 RSB ; And return.
032F 990
.END

SS.PSECT_EP	= 00000000		GBPB\$B_BID	= 00000008
SSRMSTEST	= 0000001A		GBPB\$C_BID	= 00000015
SSRMS_PBUGCHK	= 00000010		GBSB\$B_FLAGS	= 00000008
SSRMS_TBUGCHK	= 00000008		GBSB\$L_GS_SIZE	= 00000038
SSRMS_UMODE	= 00000004		GBSB\$M_NOTACCESSED	= 00000001
SST1	= 00000001		GBSB\$W_GBC	= 00000034
BDB\$B_BID	= 00000008		ID_SIZE	= 0000001C
BDB\$B_FLGS	= 0000000A		IFBS\$B_JNLFLG2	= 000000A2
BDB\$C_BID	= 0000000C		IFBS\$B_MODE	= 0000000A
BDB\$V_DRT	= 00000001		IFBS\$B_ORGCASE	= 00000023
BDB\$W_USERS	= 0000000C		IFBS\$C_IDX	= 00000002
BKP	= 00000020		IFBS\$L_ARGLST	= 00000018
BLB\$L_LOCK_ID	= 00000024		IFBS\$L_ASBBADDR	= 00000014
BLDFIB	0000008E R 01		IFBS\$L_BDB_FLNK	= 00000040
CHKGBL	00000291 R 01		IFBS\$L_BLBBLNK	= 0000009C
CLNJNL	00000153 R 01		IFBS\$L_BLBFLNK	= 00000098
CLSCU1	000000BE R 01		IFBS\$L_FWA_PTR	= 00000038
CLSDLT	00000073 R 01		IFBS\$L_GBH_PTR	= 00000088
CLSMAGTAP	00000123 R 01		IFBS\$L_GBSB_PTR	= 0000007C
CLSMMLBX	0000013E R 01		IFBS\$L_IOS	= 0000000C
CLS_EX	000000D3 R 01		IFBS\$L_IRAB_LNK	= 0000001C
DEACCESS	000000F5 R 01		IFBS\$L_NWA_PTR	= 0000003C
DEASSIGN	00000148 R 01		IFBS\$L_PRIM_DEV	= 00000000
DEASSIGN_ALT	00000145 R 01		IFBS\$V_ACCESSION	= 00000025
DELETE	0000009F R 01		IFBS\$V_DLTT	= 00000028
DEV\$V_FOR	= 00000018		IFBS\$V_DMO	= 00000028
DEV\$V_MBX	= 00000014		IFBS\$V_JNL	= 00000001
DEV\$V_NET	= 0000000D		IFBS\$V_NORECLK	= 00000033
DEV\$V_RND	= 0000001C		IFBS\$V_PPF_IMAGE	= 00000022
DEV\$V_SQD	= 00000005		IFBS\$V_RWC	= 00000027
DONE	00000308 R 01		IFBS\$V_SCF	= 0000002A
DRTBUG	000002A2 R 01		IFBS\$V_SPL	= 00000029
ERRBUG	000001B2 R 01		IFBS\$V_TMP	= 00000035
FAB\$C_BLN	= 00000050		IFBS\$V_WRTACC	= 00000030
FAB\$C_SEQ	= 00000000		IFBS\$W_AVGBPB	= 00000086
FAB\$L_FOP	= 00000004		IFBS\$W_AVLCL	= 00000084
FAB\$L_STV	= 0000000C		IFBS\$W_CHNL	= 00000020
FAB\$V_DLTT	= 0000000F		IMPSC_ASYQIOEFN	= 0000001F
FAB\$V_DMO	= 0000000C		IOSM_DELETE	= 00000100
FAB\$V_RWC	= 0000000B		IOS_DELETE	= 00000035
FAB\$V_SCF	= 0000000E		IRBS\$B_MODE	= 0000000A
FAB\$V_SPL	= 0000000D		IRBS\$L_BKPBITS	= 00000004
FAB\$WIFI	= 00000002		IRBS\$L_IRAB_LNK	= 0000001C
FIB\$V_FINDFID	= 0000000B		IRBS\$V_ASYNC	= 00000023
FIB\$W_DID	= 0000000A		IRBS\$V_BUSY	= 00000020
FIB\$W_FID	= 00000004		IRBS\$V_PPF_IMAGE	= 00000022
FIB\$W_NMCTL	= 00000014		MASK	= 000001FF
FOP	= 00000020		NTSNWA_FREE	***** X 01
FTLS DEALLERR	= FFFFFFFE		NXTDISC	0000000D R 01
FTLS RLSDRT	= FFFFFFFD8		NXTIRAB	00000048 R 01
FWASQ_FIB	= 00000010		NXTTRTN	00000036 R 01
FWASQ_SHRFIL	= 00000190		PIOSA_TRACE	***** X 01
FWAST_FIBBUF	= 000001F4		RAB\$C_BLN	= 00000044
GBDSL_LOCK_ID	= 00000014		RAB\$L_STV	= 0000000C
GBHSL_GS_SIZE	= 00000010		RAB\$W_ISI	= 00000002
GBHSL_LOCK_ID	= 00000014		RA_EX	0000029E R 01
GBHSL_USECNT	= 0000001C		RETIFB	00000162 R 01

RMSOCLOSE
Symbol table

DISPATCH FOR CLOSE OPERATION

N 6

16-SEP-1984 01:11:09 VAX/VMS Macro V04-00
5-SEP-1984 16:24:38 [RMS.SRC]RMSOCLOSE.MAR;1

Page 26
(14)

RLS_FILE_LOCK
RM\$BUG
RM\$CLEANUP
RM\$CLOSE3
RM\$CLSCU
RM\$DEACCESS
RM\$DEAJNL
RM\$DEALLOCATE_FWA
RM\$DISCOMMONSOC
RM\$DISCONNECT1
RM\$DISCONNECT3
RMSEXRMS
RMSEX_NOSTR
RM\$FCFFNC
RM\$FSET
RM\$MAPJNL
RM\$RAISE_GBS_LOCK
RM\$RELEASEALL
RM\$RELEASE_GBL_BUFFERS
RM\$RET1PAG
RM\$RETBDB
RM\$RETBBL
RM\$RETBLL
RM\$RETGBPB
RM\$RETIFB
RM\$REWIND_MT
RM\$RLNER1
RM\$RLNERR
RM\$RLS_GBSB
RM\$RLS_SFSB
RM\$SPL_SCF
RM\$UNMAP_GBL
RM\$UNMAP_GBL_ALT
RM\$WRITEOF
RM\$WTAPMARK
RM\$ZAPIFI
RM\$CLOSE
RM\$SGW_GBLBUFQUO
RM\$MRD
RM\$SPL
SJCS_DELETE_FILE
SJCS_ENTER_FILE
SJCS_FILE_IDENTIFICATION
SJCS_QUEUE
SYSSDASSGN
SYSSDELTV
SYSSDEQ
SYSSNDJBC
SYSBATCH
SYSBATCH_LEN
SYSPRINT
SYSPRINT_LEN
TPTSL_CLOSE

000002F0 R 01
***** X 01
000000DE RG 01
***** X 01
000000BC RG 01
***** X 01
0000023D RG 01
000002A9 RG 01
***** X 01
000000D6 RG 01
***** X 01
000001CB RG 01
0000030C RG 01
00000318 RG 01
***** X 01
***** X 01
***** X 01
***** X 01
= FFFFFFFE RG 01
***** X 01
= 0001C032
= 0001C042
= 00000018
= 00000013
= 00000027
= 00000086
***** GX 01
***** GX 01
***** GX 01
***** GX 01
000001C2 R 01
= 00000009
000001B9 R 01
= 00000009
***** X 01

RM
VO

```
+-----+
! Psect synopsis !
+-----+
```

PSECT name

	Allocation	PSECT No.	Attributes																
ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE																
RMSRMS	0000032F (815.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE																
SABSS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE																

```
+-----+
! Performance indicators !
+-----+
```

Phase

	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.09	00:00:00.95
Command processing	110	00:00:00.71	00:00:04.85
Pass 1	593	00:00:25.51	00:01:00.17
Symbol table sort	0	00:00:03.95	00:00:05.79
Pass 2	173	00:00:04.85	00:00:09.28
Symbol table output	21	00:00:00.22	00:00:00.74
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	932	00:00:35.36	00:01:21.91

The working set limit was 1950 pages.

140168 bytes (274 pages) of virtual memory were used to buffer the intermediate code.

There were 140 pages of symbol table space allocated to hold 2678 non-local and 29 local symbols.

991 source lines were read in Pass 1, producing 17 object records in Pass 2.

48 pages of virtual memory were used to define 47 macros.

```
+-----+
! Macro library statistics !
+-----+
```

Macro library name

	Macros defined
\$255\$DUA28:[RMS.OBJ]RMS.MLB;1	24
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	2
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	17
TOTALS (all libraries)	43

2890 GETS were required to define 43 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:RMSOCLOSE/OBJ=OBJ\$:RMSOCLOSE MSRC\$:RMSOCLOSE/UPDATE=(ENH\$:RMSOCLOSE)+EXECML\$/LIB+LIB\$:RMS/LIB

0329 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

RMS0XHELD LIS	RMS0CONN LIS	RMS0CREAT LIS	RMS0DELET LIS	RMS0ENTER LIS	RMS0ERASE LIS	RMS0FSCHN LIS
RMS0XSUMO LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0FIND LIS
RMS0KEYO LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0DISPL LIS	RMS0GET LIS
RMS0BLKIO LIS	RMS0CLOSE LIS	RMS0DISC LIS	RMS0DISC LIS	RMS0EXTEN LIS	RMS0EXTEN LIS	RMS0EXTEN LIS